



Programación Python para BigData

Lección 5: Base de datos: MongoDB

ÍNDICE

Lección 5: Bases de datos: MongoDB	2
Presentación y objetivos.....	2
1. MongoDB.....	3
2. Robo3T.....	9
3. Pymongo.....	18
4. Puntos claves	25

Lección 5: Bases de datos: MongoDB

PRESENTACIÓN Y OBJETIVOS

En esta lección aprenderemos a trabajar con un ejemplo de base de datos No relacionales como es MongoDB, usando tres instrumentos para saber cómo leer, insertar, actualizar o eliminar un dato. (CRUD)



Objetivos

- | Trabajar a través de consola usando directamente MongoDB
- | Trabajar con Robo3T es un visualizador de la base de datos a través de aplicación GUI
- | Trabajar con una librería de Python como es Pymongo

1. MONGODB

Introducción

MongoDB es una base de datos NoSQL o también llamada No relacionales, orientada a documentos. A diferencia de las bases de datos relacionales, donde se almacenan los registros como filas de una tabla, con MongoDB estos registros son almacenados en documentos BSON (un derivado de los documentos JSON).

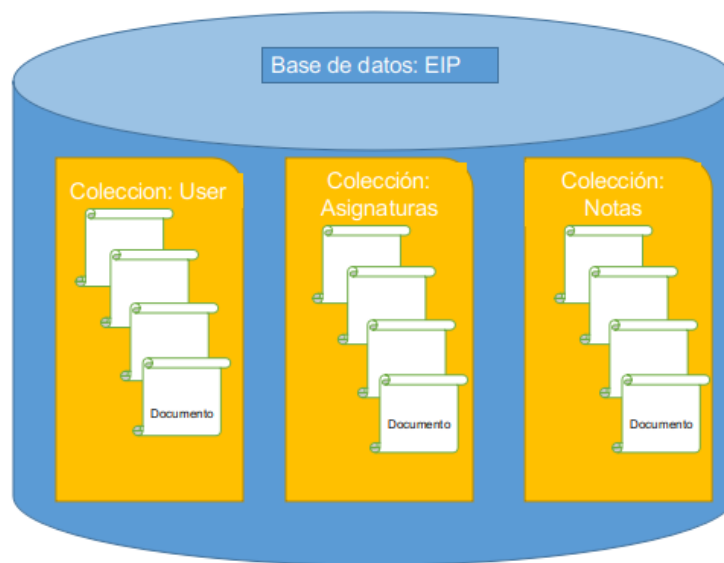


Figura 1.1 Imagen de la estructura que tiene MongoDB

Presenta las siguientes ventajas:

- | Uso de esquema flexible: No existe esquema, por lo que cada documento podría tener una estructura distinta
- | Escalabilidad horizontal: se puede crear un clúster donde la información estará particionada y replicada lo que permite poder escalar nuestro sistema horizontalmente en base a la demanda
- | Software Open Source: es una base de datos de código abierto y gratuita con una gran comunidad muy involucrada

Para trabajar con ello usaremos distintas herramientas como es:

- | Usar directamente **MongoDB** a través de instrucciones de consola.

- | Usar **Robo3T** que es una aplicación GUI para realizar esas mismas instrucciones, además nos permite conectarnos a varias bases de datos a la vez.
- | Usar **pymongo** una librería de Python que nos permite conectarnos a la base de datos y realizar todas las instrucciones.

Realizaremos las actividades descritas como **CRUD**, que viene del Inglés (Create, Read, Update and Delete):

Create / Crear: crearemos un nuevo dato en nuestra colección.

Read / leer: leeremos los datos de la colección.

Update / actualizar: actualizaremos un dato ya existente en la colección.

Delete / eliminar: eliminamos un dato de la colección.

MongoDB instrucciones:

Para trabajar con MongoDB necesitamos usar el docker realizado en la actividad de la Lección 2, deberemos de tener el proceso activo:

```
isabel@isabel-SVE1512E1EW:~$ sudo docker ps
CONTAINER ID   IMAGE      COMMAND                  CREATED        STATUS        PORTS                               NAMES
1cc592f8f6d0   mongo     "docker-entrypoint.s..." 28 seconds ago Up 27 seconds 0.0.0.0:27017->27017/tcp, :::27017->27017/tcp mongodb
isabel@isabel-SVE1512E1EW:~$
```

Figura 1.2 Podemos ver la imagen docker en ejecución de MongoDB

Entramos en la imagen del servicio MongoDB:

```
sudo docker exec -it mongodb bash
```

```
isabel@isabel-SVE1512E1EW:~$ sudo docker exec -it mongodb bash
root@1cc592f8f6d0:/#
```

Figura 1.3 Ejecución de MongoDB

Ponemos mongo para entrar:

```
root@1cc592f8f6d0:/# mongo
```

```
isabel@isabel-SVE1512E1E1E1:~$ sudo docker exec -it mongodb bash
root@1cc592f8f6d0:/# mongo
MongoDB shell version v4.4.6
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("4f956c25-b0de-4218-a716-1afa5148f266") }
MongoDB server version: 4.4.6
---
The server generated these startup warnings when booting:
  2021-07-08T16:28:46.556+00:00: Using the XFS filesystem is strongly recommended with the WiredTiger storage engine. See http://dochub.mongodb.org/core/prodnotes-filesystem
  2021-07-08T16:28:47.396+00:00: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
---
---
  Enable MongoDB's free cloud-based monitoring service, which will then receive and display
  metrics about your deployment (disk utilization, CPU, operation statistics, etc).

  The monitoring data will be available on a MongoDB website with a unique URL accessible to you
  and anyone you share the URL with. MongoDB may use this information to make product
  improvements and to suggest MongoDB products and deployment options to you.

  To enable free monitoring, run the following command: db.enableFreeMonitoring()
  To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
---
>
```

Figura 1.4 Ejecución de MongoDB

Para ver las bases de datos con mongodb ponemos:

```
show dbs
```

```
root@1cc592f8f6d0:/# mongo
MongoDB shell version v4.4.6
connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb
Implicit session: session { "id" : UUID("4f956c25-b0de-4218-a716-1afa5148f266") }
MongoDB server version: 4.4.6
---
The server generated these startup warnings when booting:
  2021-07-08T16:28:46.556+00:00: Using the XFS filesystem is strongly recommended with the WiredTiger storage engine. See http://dochub.mongodb.org/core/prodnotes-filesystem
  2021-07-08T16:28:47.396+00:00: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
---
---
  Enable MongoDB's free cloud-based monitoring service, which will then receive and display
  metrics about your deployment (disk utilization, CPU, operation statistics, etc).

  The monitoring data will be available on a MongoDB website with a unique URL accessible to you
  and anyone you share the URL with. MongoDB may use this information to make product
  improvements and to suggest MongoDB products and deployment options to you.

  To enable free monitoring, run the following command: db.enableFreeMonitoring()
  To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
---
> show dbs
admin    0.000GB
config  0.000GB
local    0.000GB
>
```

Figura 1.5 Mostrar las bases de datos

Insertar datos

Para crear una base de datos nueva será necesario insertar un dato nuevo:

use eip ---> nombre de la BBDD

```
db.user.insert({name: "Isabel Maniega", age: 205})
```

Donde:

db: siempre se pone para referir a base de datos

User: colección de datos que vamos a crear

.insert({}): insertar un dato nuevo

```
> show dbs
admin    0.000GB
config  0.000GB
local    0.000GB
> use eip
switched to db eip
> db.user.insert({"name":"Isabel Maniega", "age":205})
WriteResult({ "nInserted" : 1 })
> █
```

Figura 1.6 Insertar un dato para crear la base de datos "eip"

Volvemos a poner:

```
show dbs
```

```
> show dbs
admin    0.000GB
config  0.000GB
eip      0.000GB
local    0.000GB
> █
```

Figura 1.7 Base de datos "eip"

Aparece la base de datos que acabamos de crear "eip"

Para acceder a la colección ponemos:

```
use eip

show collections
```

```
> use eip
switched to db eip
> show collections
user
> █
```

Figura 1.8 Base de datos "eip"

Vemos que tenemos la colección "user".

Actualización de dato

Para realizar la actualización de un dato realizamos la siguiente instrucción:

```
db.user.update({_id:ObjectId("60ec1c4c04dae9c1c5c3a1ed")}, {"name": "Isabel Maniega", "age":210})
```

```
> db.user.update({_id:ObjectId("60ec1c4c04dae9c1c5c3a1ed")}, {"name": "Isabel Maniega", "age":210})
WriteResult({ "nMatched" : 1, "nUpserted" : 0, "nModified" : 1 })
```

Figura 1.9 Actualización de un dato

Donde:

db.user.update({}) --> instrucción para actualizar un dato.

Primer json: Se pondrá el _id del dato a actualizar.

Segundo json: el json del dato actualizado.

Para mostrar los datos:

```
db.user.find({})
```

```
> db.user.find({})
{ "_id" : ObjectId("60ec1c4c04dae9c1c5c3a1ed"), "name" : "Isabel Maniega", "age" : 210 }
{ "_id" : ObjectId("60ec21b9e0a32d8d2ec13e71"), "name" : "José Manuel Peña", "age" : 200 }
```

Figura 1.10 Lectura de los datos

Eliminar un dato

```
db.user.deleteOne({_id:ObjectId("60ec1c4c04dae9c1c5c3a1ed")})
```

Pondremos la instrucción deleteOne({}) + el _id del dato a eliminar:

```
> db.user.deleteOne({_id:ObjectId("60ec1c4c04dae9c1c5c3a1ed")})
{ "acknowledged" : true, "deletedCount" : 1 }
```

Figura 1.11 Eliminación de un dato

Comprobamos que se ha eliminado:


```
db.user.find({})
```

```
> db.user.find({})  
{ "_id" : ObjectId("60ec21b9e0a32d8d2ec13e71"), "name" : "José Manuel Peña", "age" : 200 }
```

Figura 1.12 Lectura de los datos

2. ROBO3T

Es una aplicación GUI que nos permite conectarnos a cualquier Base de datos MongoDB a través de la IP y Port, y sus credenciales, así como realizar su gestión.

Para conectar con la base de datos MongoDB en local ponemos:

File --> connect o pulsamos el icono de los ordenadores esquina superior izquierda:

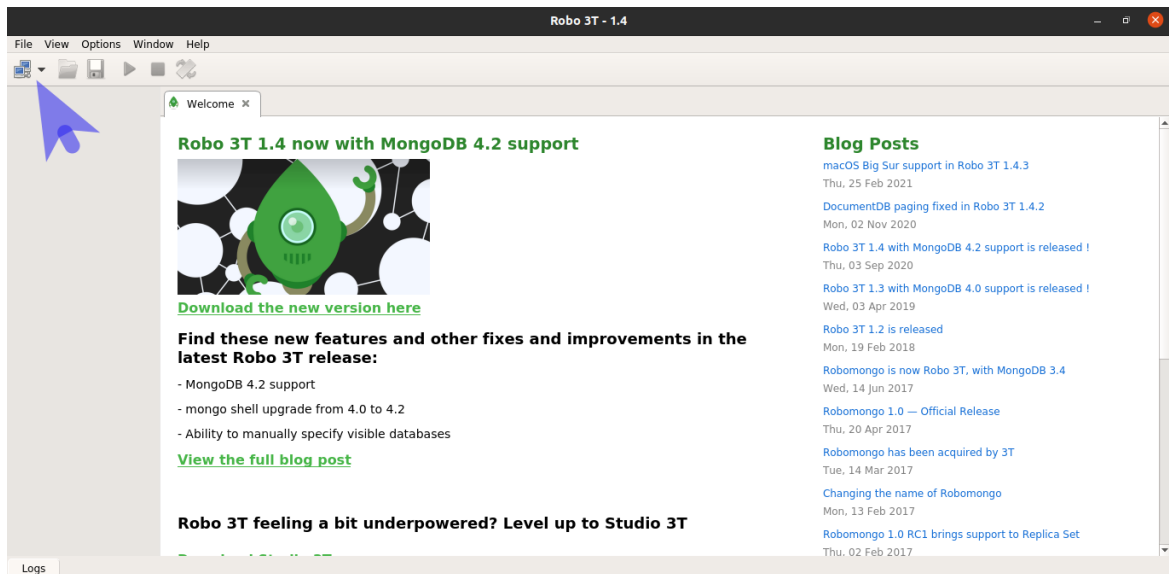


Figura 2.1 Conexión a mongodb en localhost

Damos en create:

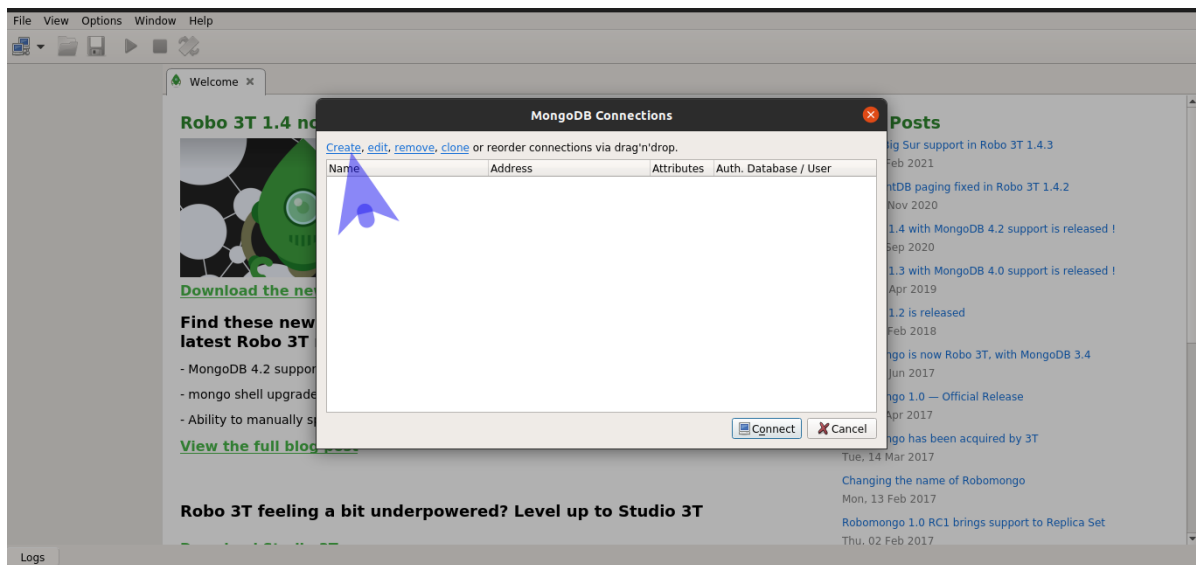


Figura 2.2 Conexión a mongodb en localhost

Ponemos a la conexión:

Name: mongoDocker

Adress: localhost:27017

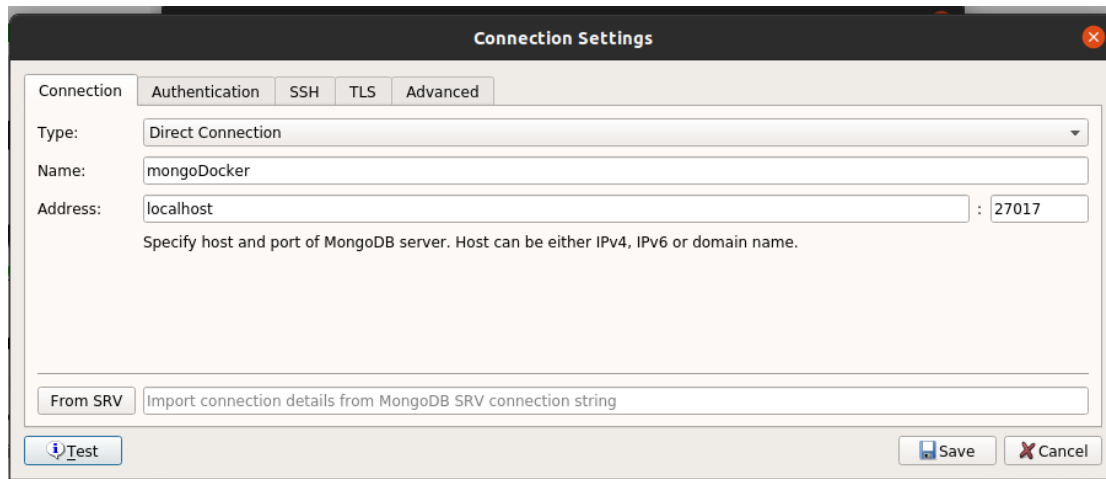


Figura 2.3 Conexión a mongodb en localhost

Damos a test para comprobar si es correcto:

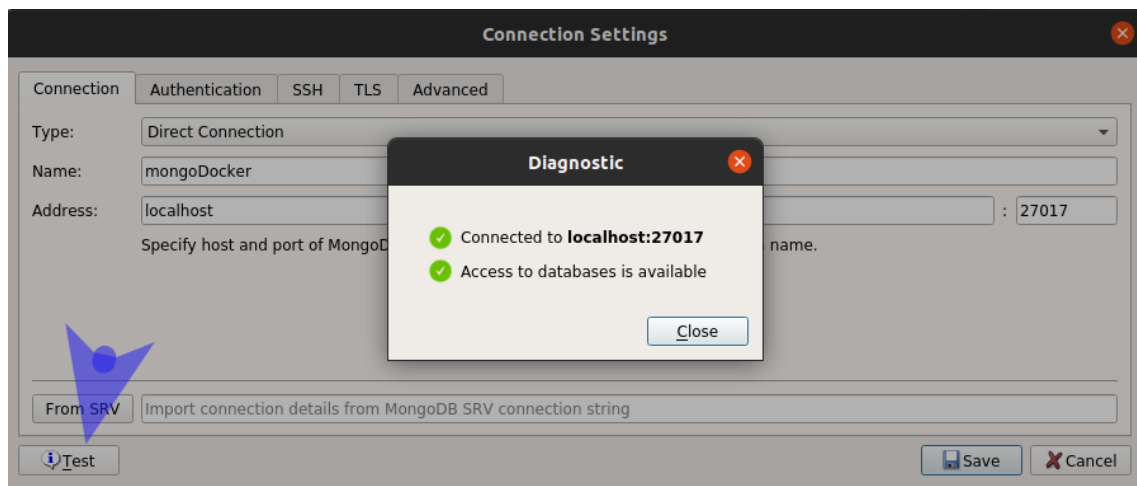


Figura 2.4 Comprobación de la conexión a mongodb en localhost

Vemos que es correcto y damos a “save” para guardar.

Ya tenemos robo3t conectada a mongodb.

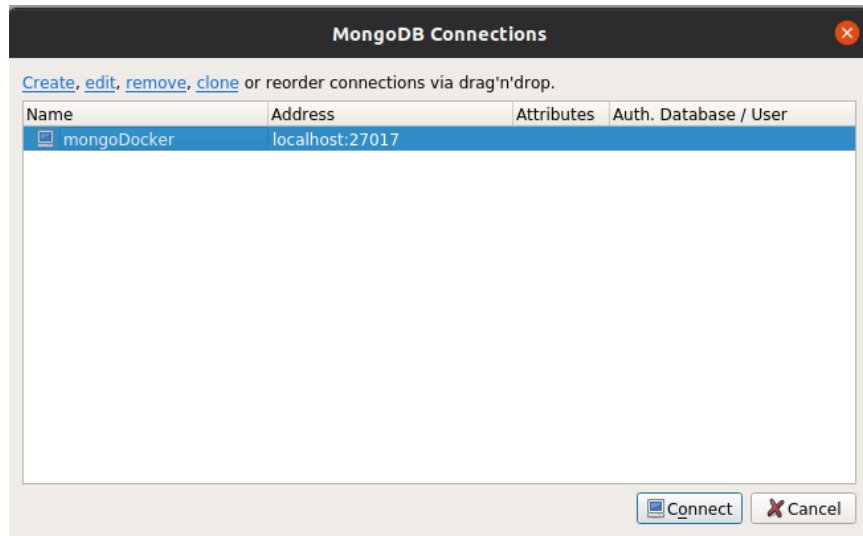


Figura 2.5 Conexión a mongodb en localhost

Damos a “connect” y nos muestra nuestra base de datos:

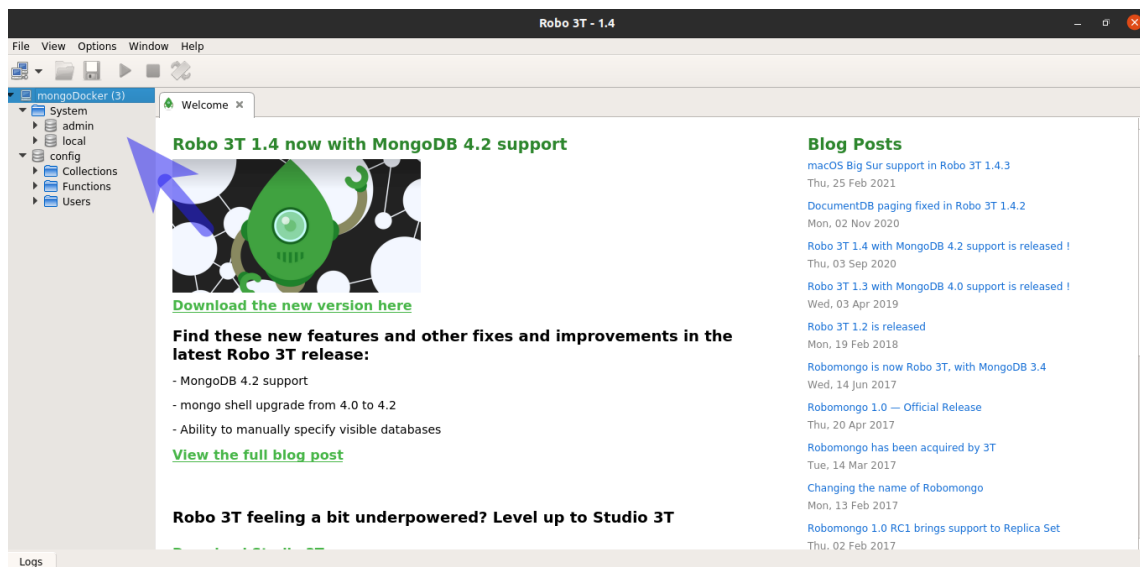


Figura 2.6 Conexión a mongodb en localhost

Crear base de datos con Robo3T

Para crear una base de datos nos ponemos encima del nombre de la base de datos --> Botón derecho --> seleccionamos crear base de datos (create database):

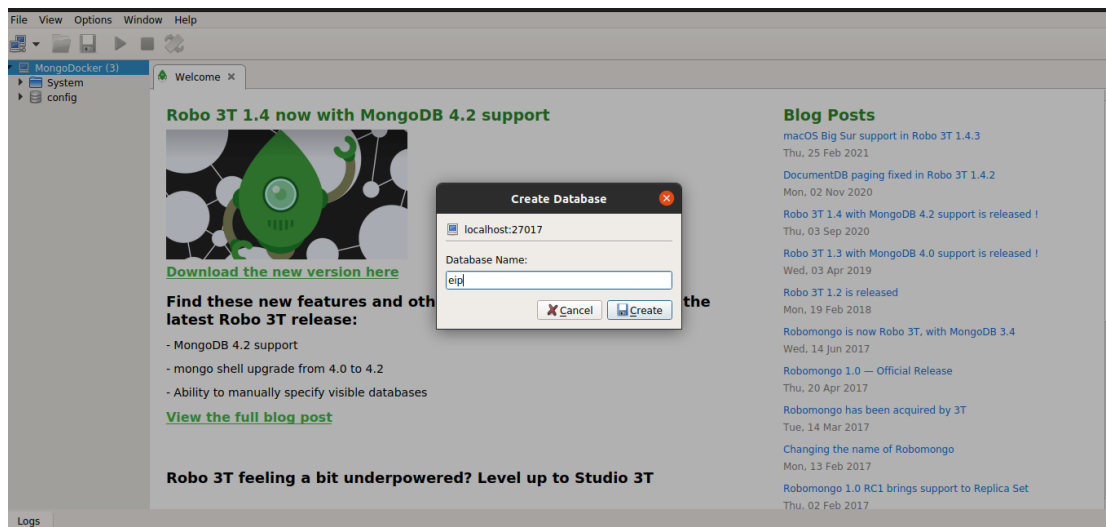


Figura 2.7 Conexión a mongodb en localhost

Pondremos de nombre “eip” y daremos a create:

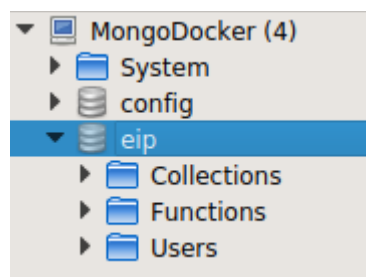


Figura 2.8 Muestra la base de datos “eip”

A continuación creamos una colección de datos nueva, nos colocamos encima de collections --> crear colección (create collection) :

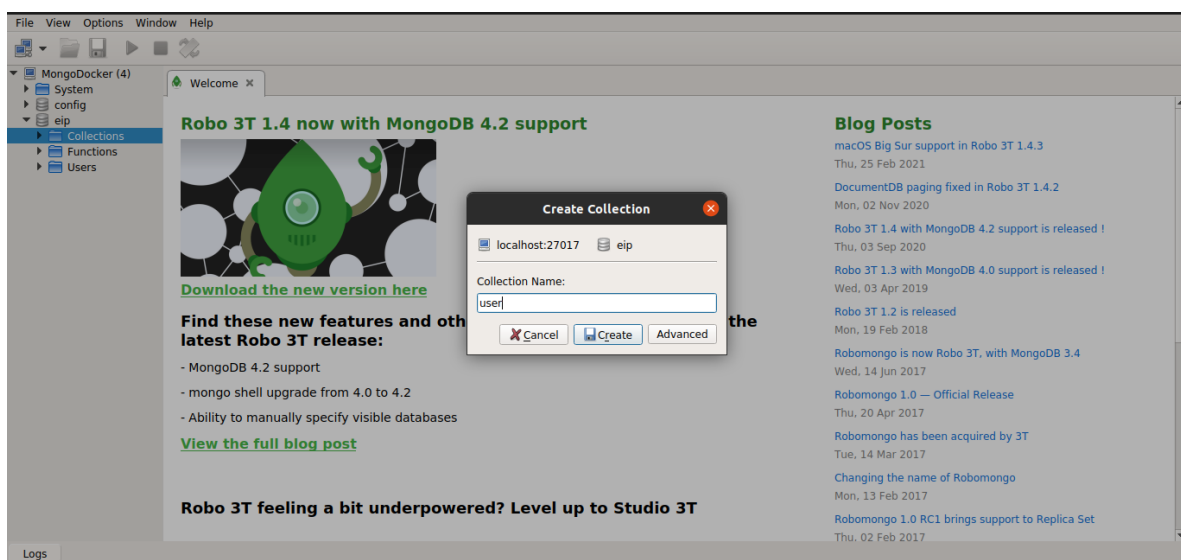


Figura 2.9 Crear colección de datos “user”

Ponemos de nombre "user" y pulsamos create:

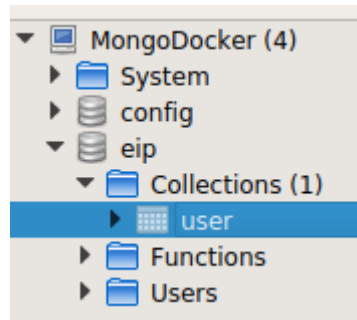


Figura 2.10 Crear colección de datos "user"

Insertar un dato

Para crear un dato nuevo nos colocamos en la colección creada "user" --> botón derecho --> insert document, nos muestra una ventana con formato json, donde pondremos nuestros datos:

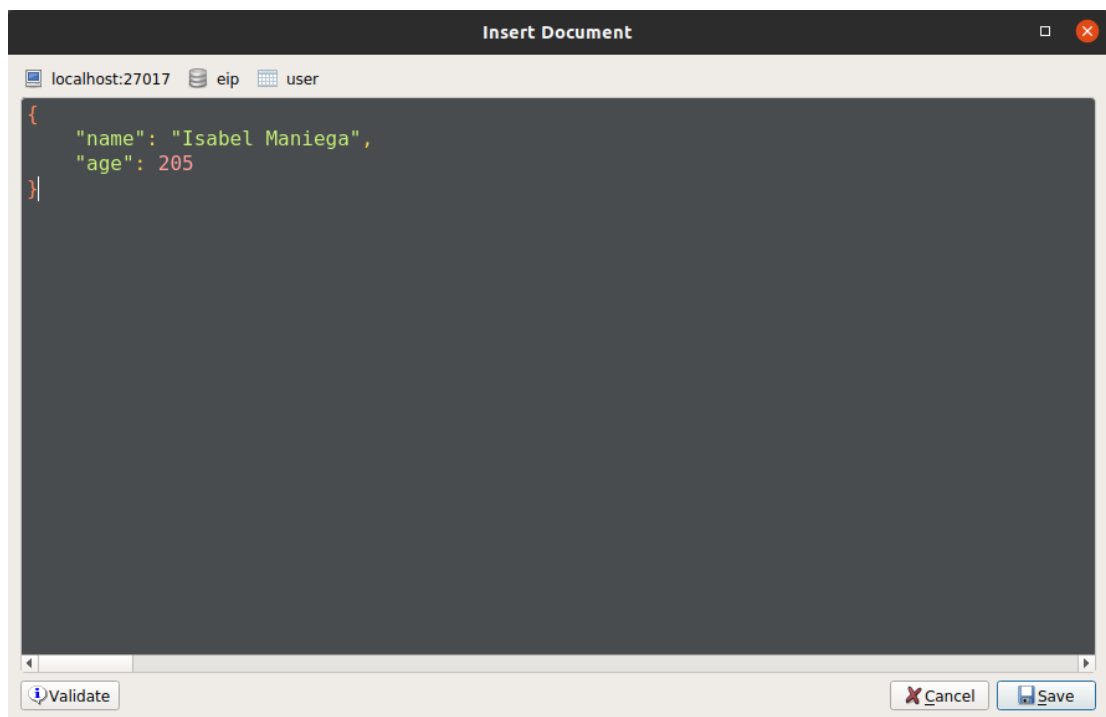


Figura 2.11 Insertar un dato en la colección "user"

Ponemos el siguiente json:

```
{  "name": "Isabel Maniega",  "age": 205}
```

Pulsamos save.

Si pulsamos dos veces encima de la colección de nombre “user”, nos muestra el dato que acabamos de insertar:

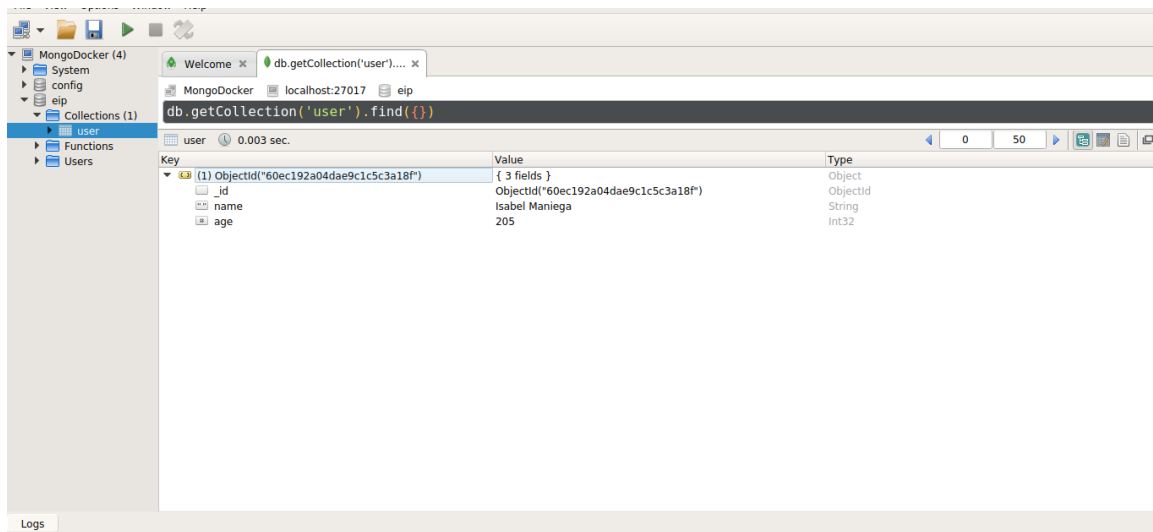


Figura 2.12 Mostrar datos de la colección “user”

Podemos ver una consola de comandos encima donde se puede filtrar/buscar datos:

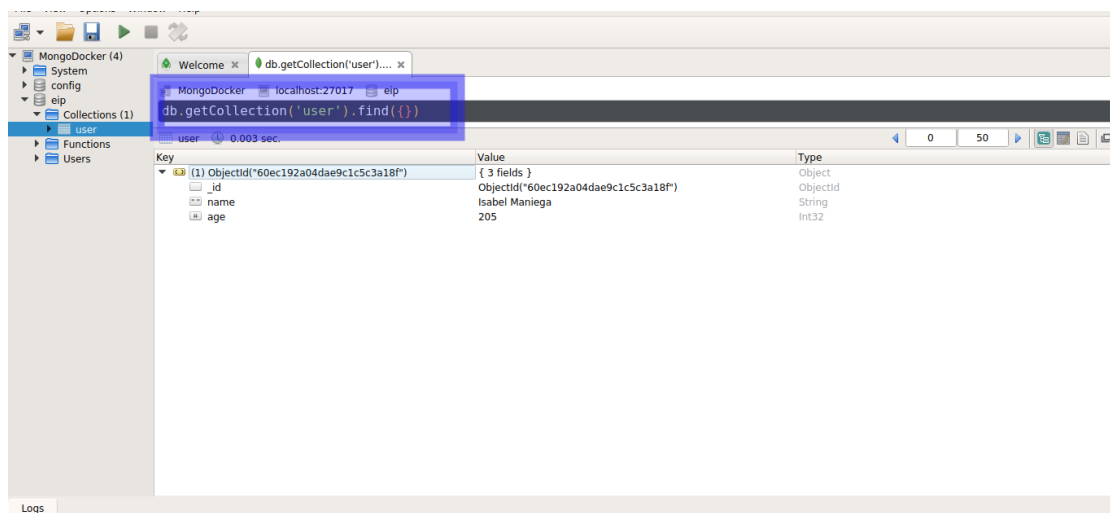


Figura 2.13 Mostrar datos de la colección “user”

Actualizar un dato

Para actualizar un dato nos pondremos encima del dato insertado anteriormente --> botón derecho --> edit document:

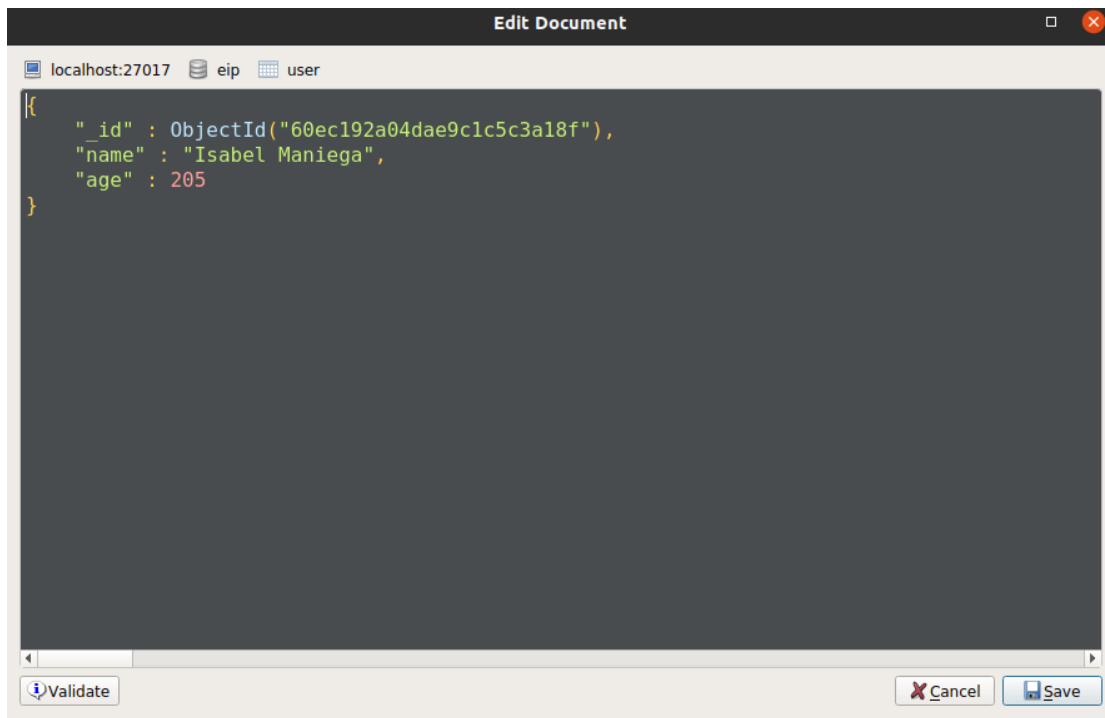


Figura 2.14 Actualizar un dato en la colección "user"

Nos muestra la ventana con el documento a modificar:

Modificamos por ejemplo la edad a 210:

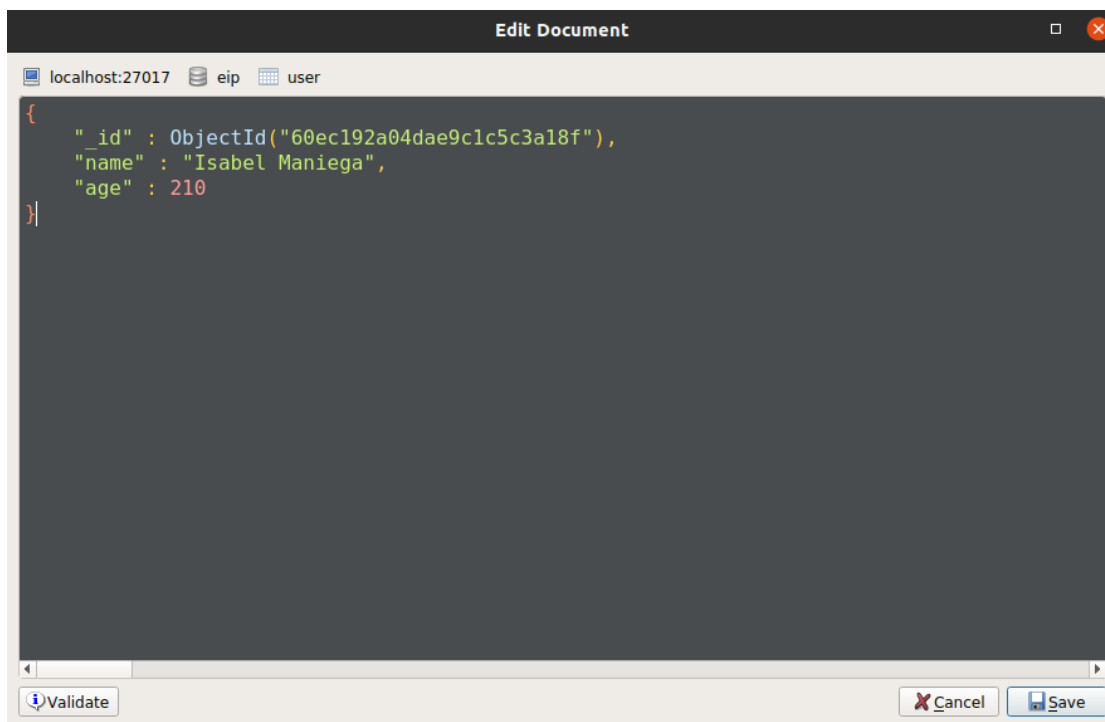


Figura 2.15 Actualizar un dato en la colección "user"

Y pulsamos save.

Key	Value	Type
(1) ObjectId("60ec192a04dae9c1c5c3a18f")	{ 3 fields }	Object
_id	ObjectId("60ec192a04dae9c1c5c3a18f")	ObjectId
name	Isabel Maniega	String
age	210	Int32

Figura 2.16 Actualizar un dato en la colección "user"

Vemos que nos ha modificado el dato.

En todas las ventanas podemos observar un botón de "validate" que nos permite ver si el JSON a insertar tiene el formato correcto:

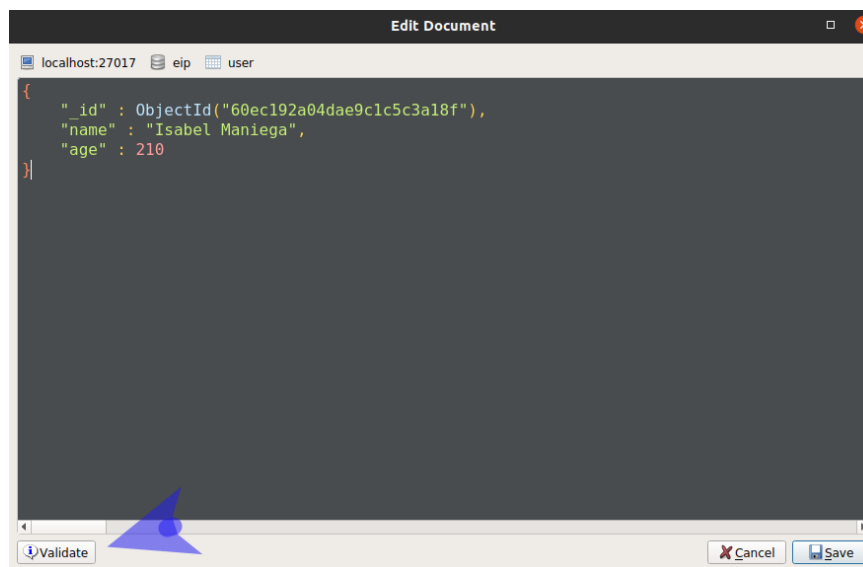


Figura 2.17 Comprobación del formato JSON

Si es así nos mostrará un mensaje de que es válido.

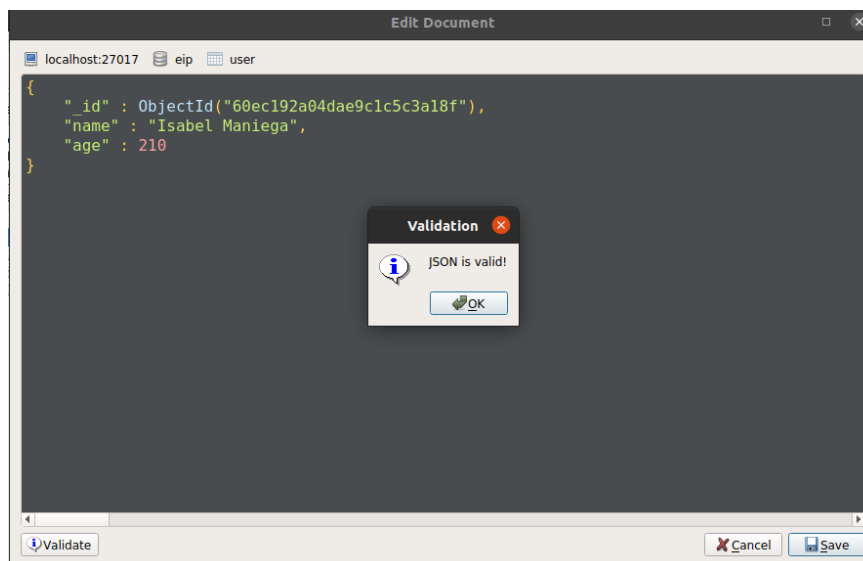


Figura 2.18 Comprobación del formato JSON

Eliminar un dato

Para ello nos colocamos igual que antes encima del dato --> botón derecho --> delete document:

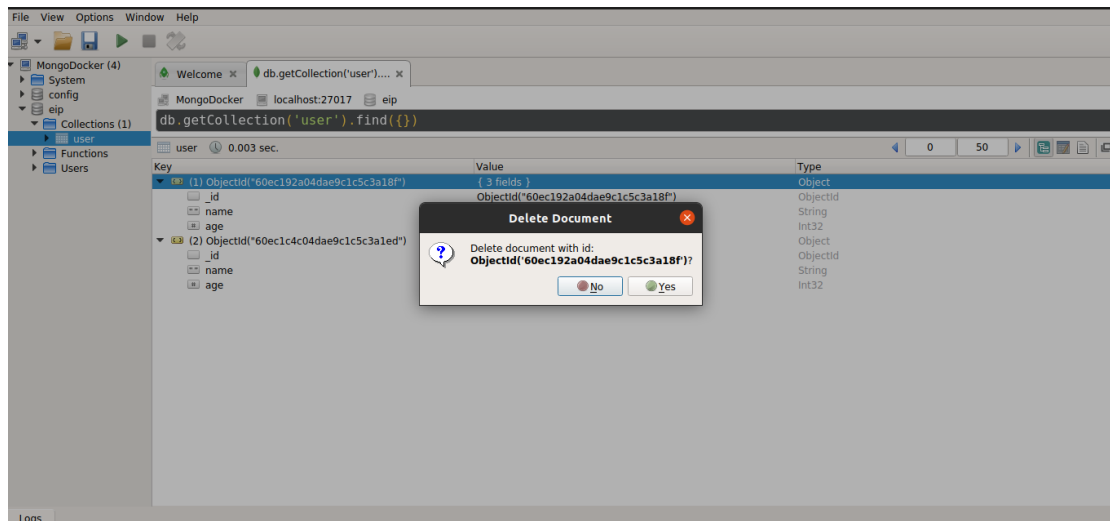


Figura 2.19 Eliminación de una datos en la colección "eip"

Nos pregunta si queremos eliminar el dato, daremos "yes" en caso de ser así o "no" en caso contrario.

Si dimos que "si", veremos cómo ya no tenemos el dato.

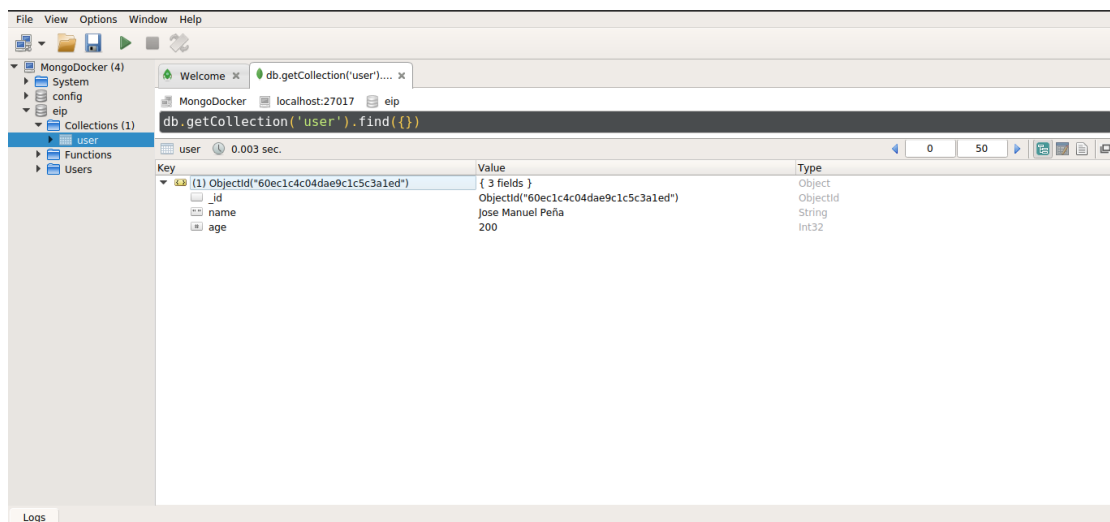


Figura 2.20 Colección "eip"

Para recargar los datos pulsaremos en el triángulo verde:



Figura 2.21 Recargar la colección "eip"

3. PYMONGO

Para conectarnos desde un script de Python a MongoDB usaremos la librería pymongo:

<https://pypi.org/project/pymongo/>

Creamos una carpeta para realizar el script de conexión a la base de datos con nombre “pymongo”:

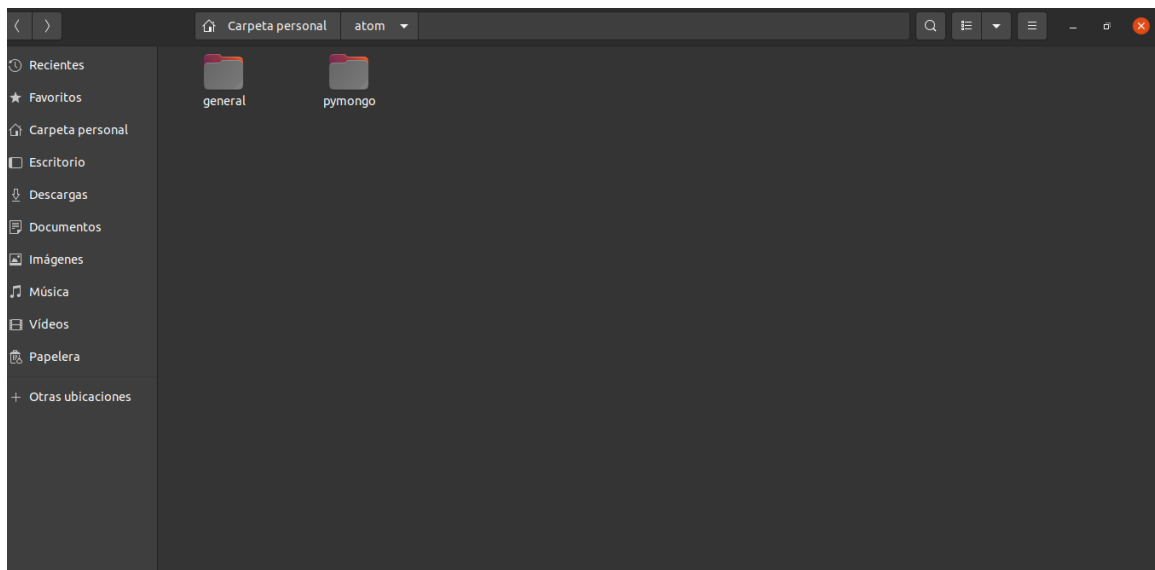


Figura 3.1 Creando el proyecto pymongo

Entramos en la carpeta donde creamos el entorno virtual con virtualenv:

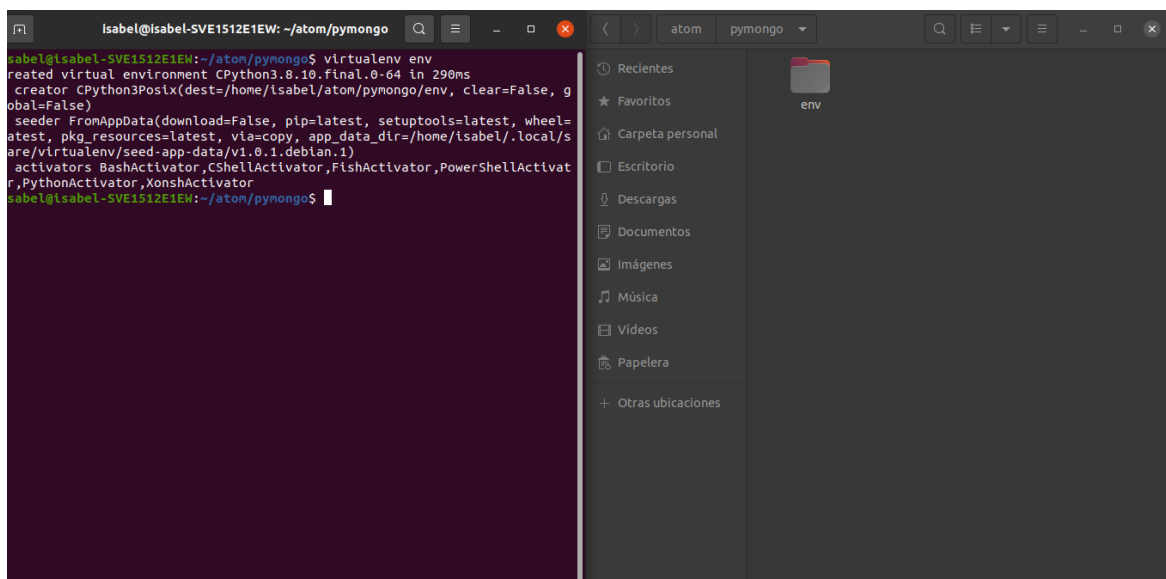
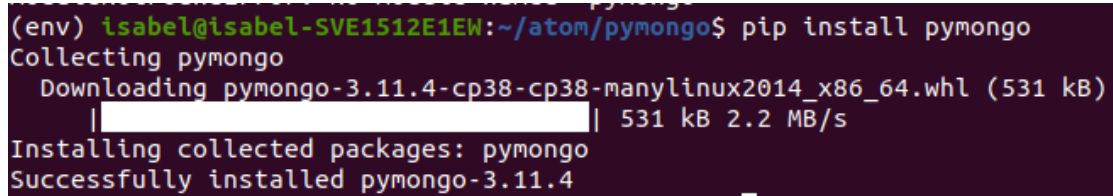


Figura 3.2 Creando el entorno virtual

Instalamos la librería de pymongo:

```
pip install pymongo
```



```
(env) isabel@isabel-SVE1512E1EW:~/atom/pymongo$ pip install pymongo
Collecting pymongo
  Downloading pymongo-3.11.4-cp38-cp38-manylinux2014_x86_64.whl (531 kB)
    | 531 kB 2.2 MB/s
Installing collected packages: pymongo
Successfully installed pymongo-3.11.4
```

Figura 3.3 Instalación de pymongo

Vamos a atom y abrimos el proyecto creado, creamos el script con nombre “mongodb.py”:

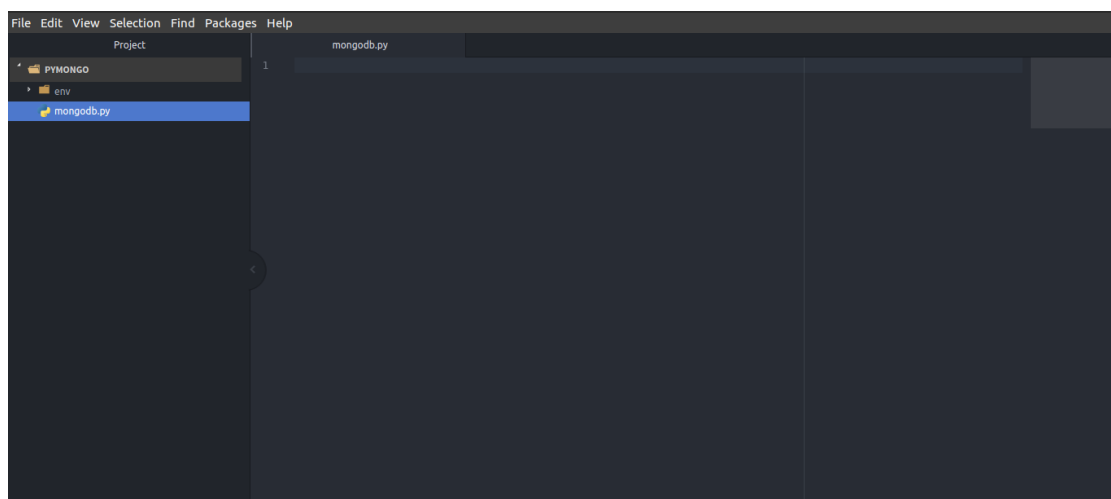


Figura 3.4 Crear el script mongodb.py

Creamos un cliente para la conexión con la base de datos poniendo:

```
import pymongo

# crearemos un cliente para conectarnos a la base de datos
# añadimos el IP (localhost) y host (27017)
client = pymongo.MongoClient("localhost", 27017)
# creamos conexión a la base de datos en nuestro caso eip
db = client.eip
# cargamos la colección de datos
print(db.user)
```

```
mongodb.py
import pymongo

# crearemos un cliente para conectarnos a la base de datos
# añadimos el IP (localhost) y host (27017)
client = pymongo.MongoClient("localhost", 27017)
# creamos conexión a la base de datos en nuestro caso eip
db = client.eip
# cargamos la colección de datos
print(db.user)
```

Figura 3.5 Conexión a mongodb

Ejecutamos:

```
python mongodb.py
```

```
(env) isabel@isabel-SVE1512E1EW:~/atom/pymongo$ python mongodb.py
Collection(Database(MongoClient(host=['localhost:27017'], document_class=dict, tz_aware=False, connect=True), 'eip'), 'user')
```

Figura 3.6 Crear el script mongodb.py

Observamos que se ha realizado correctamente la conexión a la base de datos.

Vamos a mostrar los datos que tenemos en la colección para ello:

```
# usamos la instrucción de db.user.find({})
datos = db.user.find({})
# necesitamos recorrer los datos para mostrarlos
for dato in datos:
    print(dato)
```

```
mongodb.py
1 import pymongo
2
3
4 # crearemos un cliente para conectarnos a la base de datos
5 # añadimos el IP (localhost) y host (27017)
6 client = pymongo.MongoClient("localhost", 27017)
7 # creamos conexión a la base de datos en nuestro caso eip
8 db = client.eip
9 # cargamos la colección de datos
10 print(db.user)
11
12 # usamos la instrucción de db.user.find({})
13 datos = db.user.find({})
14 # necesitamos recorrer los datos para mostrarlos
15 for dato in datos:
16     print(dato)
17
```

Figura 3.7 Leer los datos de la base de datos "eip"

Ejecutamos el script:

```
(env) isabel@isabel-SVE1512E1EW:~/atom/pymongo$ python mongodb.py
Collection(Database(MongoClient(host=['localhost:27017'], document_class=dict, tz_aware=False, connect=True), 'eip'), 'user')
{'_id': ObjectId('60ec21b9e0a32d8d2ec13e71'), 'name': 'José Manuel Peña', 'age': 200.0}
```

Figura 3.8 Leer los datos de la base de datos “eip”

Observamos en la segunda línea que nos muestra el dato que tenemos en la base de datos.

Insertar un dato

```
# insertar un nuevo dato
db.user.insert({"name": "Isabel Maniega", "age": 205})
```

Volvemos a poner que nos muestre los datos:

```
# usamos la instrucción de db.user.find({})
datos = db.user.find({})
# necesitamos recorrer los datos para mostrarlos
for dato in datos:
    print(dato)
```

```
(env) isabel@isabel-SVE1512E1EW:~/atom/pymongo$ python mongodb.py
Collection(Database(MongoClient(host=['localhost:27017'], document_class=dict, tz_aware=False, connect=True), 'eip'), 'user')
{'_id': ObjectId('60ec21b9e0a32d8d2ec13e71'), 'name': 'José Manuel Peña', 'age': 200.0}
mongodb.py:19: DeprecationWarning: insert is deprecated. Use insert_one or insert_many instead.
  db.user.insert({"name": "Isabel Maniega", "age": 205})
{'_id': ObjectId('60ec21b9e0a32d8d2ec13e71'), 'name': 'José Manuel Peña', 'age': 200.0}
{'_id': ObjectId('60ec2a7fe6199e9da9367b7e'), 'name': 'Isabel Maniega', 'age': 205}
```

Figura 3.9 Insertar y leer los datos de la base de datos “eip”

Observamos que se ha insertado correctamente.

Actualizar un dato

Usaremos la instrucción `db.user.update_one({})`, es necesario añadirle o pasarle el `_id` que le asocia MongoDB por defecto:

Ponemos en el script:

```
# actualizar un dato:
datos = db.user.find({})
# necesitamos recorrer los datos para mostrarlos
for dato in datos:
    # Buscaremos el dato a modificar:
    if dato["name"] == "Isabel Maniega":
        # creamos una variable idMongo para recoger el id
        print(dato["_id"])
        idMongo = dato["_id"]
        # actualizamos el dato:
        db.user.update_one({"_id": idMongo},
                           {"$set": {"name": "Isabel Maniega", "age": 215}})
```

\$set: pasaremos el diccionario con los datos a modificar.

```

mongodb.py
38 # {"$set": {"name": "Isabel Maniega", "age": 215}}
39
40 # actualizar un dato:
41 datos = db.user.find({})
42 # necesitamos recorrer los datos para mostrarlos
43 for dato in datos:
44     # Buscaremos el dato a modificar:
45     if dato["name"] == "Isabel Maniega":
46         # creamos una variable idMongo para recoger el id
47         print(dato["_id"])
48         idMongo = dato["_id"]
49         # actualizamos el dato:
50         db.user.update_one({"_id": idMongo},
                             {"$set": {"name": "Isabel Maniega", "age": 215}})
51
52 # usamos la instrucción de db.user.find({})
53 datos = db.user.find({})
54 # necesitamos recorrer los datos para mostrarlos
55 for dato in datos:
56     print(dato)
57
58

```

Figura 3.10 Actualizar y leer los datos de la base de datos "eip"

```

(env) isabel@isabel-5VE1512E1EW:~/atom/pymongo$ python mongodb.py
Collection(Database(MongoClient(host=['localhost:27017'], document_class=dict, tz_aware=False, connect=True), 'eip'), 'user')
60ec2a7fe6199e9da9367b7e
{'_id': ObjectId('60ec21b9e0a32d8d2ec13e71'), 'name': 'José Manuel Peña', 'age': 200.0}
{'_id': ObjectId('60ec2a7fe6199e9da9367b7e'), 'name': 'Isabel Maniega', 'age': 215}

```

Figura 3.11 Actualizar y leer los datos de la base de datos "eip"

Observamos que lo ha actualizado correctamente.

Eliminar un dato

Usaremos la instrucción `db.user.delete_one({})`, es necesario añadirle o pasarle el `_id` que le asocia MongoDB por defecto:

```

# eliminar un dato:
datos = db.user.find({})
# necesitamos recorrer los datos para mostrarlos
for dato in datos:
    # Buscaremos el dato a modificar:
    if dato["name"] == "Isabel Maniega":
        # creamos una variable idMongo para recoger el id
        print(dato["_id"])
        idMongo = dato["_id"]
        # eliminar el dato:
        db.user.delete_one({"_id": idMongo})

```

```

38 # { $set : { name : 'Isabel Maniega', age : 213}}
39
40 # eliminar un dato:
41 datos = db.user.find({})
42 # necesitamos recorrer los datos para mostrarlos
43 for dato in datos:
44     # Buscaremos el dato a modificar:
45     if dato["name"] == "Isabel Maniega":
46         # creamos una variable idMongo para recoger el id
47         print(dato["_id"])
48         idMongo = dato["_id"]
49         # eliminar el dato:
50         db.user.delete_one({"_id": idMongo})
51
52 # usamos la instrucción de db.user.find({})
53 datos = db.user.find({})
54 # necesitamos recorrer los datos para mostrarlos
55 for dato in datos:
56     print(dato)
57

```

Figura 3.12 Eliminar y leer los datos de la base de datos "eip"

```

(env) isabel@isabel-SVE1512E1EW:~/atom/pymongo$ python mongodb.py
Collection(Database(MongoClient(host=['localhost:27017'], document_class=dict, tz_aware=False, connect=True), 'eip'), 'user')
60ec2a7fe6199e9da9367b7e
{'_id': ObjectId('60ec21b9e0a32d8d2ec13e71'), 'name': 'José Manuel Peña', 'age': 200.0}

```

Figura 3.13 Eliminar y leer los datos de la base de datos "eip"

Observamos que ya no tenemos en la base de datos al usuario Isabel.

Buscar un dato

Para buscar en la instrucción `find({})` que nos muestra todos los datos añadimos el dato que queremos buscar, usaremos el siguiente ejemplo:

```

# usamos la instrucción de db.user.find({}) para encontrar datos
datos = db.user.find({"age":203})
# necesitamos recorrer los datos para mostrarlos
for dato in datos:
    print(dato)

```



```
mongodb.py
1  import pymongo
2
3
4  # crearemos un cliente para conectarnos a la base de datos
5  # añadimos el IP (localhost) y host (27017)
6  client = pymongo.MongoClient("localhost", 27017)
7  # creamos conexión a la base de datos en nuestro caso eip
8  db = client.eip
9  # cargamos la colección de datos
10 print(db.user)
11
12 # usamos la instrucción de db.user.find({})
13 datos = db.user.find({"age":203})
14 # necesitamos recorrer los datos para mostrarlos
15 for dato in datos:
16     print(dato)
17
```

Figura 3.14 Buscar los datos de la base de datos "eip"

```
(env) isabel@isabel-SVE1512E1EW:~/atom/pymongo$ python mongodb.py
Collection(Database(MongoClient(host=['localhost:27017'], document_class=dict, tz_aware=False, connect=True), 'eip'), 'user')
{'_id': ObjectId('60ec2ec404dae9c1c5c3a409'), 'name': 'Juan Fernandez', 'age': 203.0}
```

Figura 3.15 Buscar los datos de la base de datos "eip"

4. PUNTOS CLAVES



No te olvides...

- | MongoDB es un base de datos de tipo No relacional.
- | Para visualizar los datos de manera fácil usaremos Robo3T.
- | Pymongo es una librería de Python que nos facilita la gestión de la base de datos.

